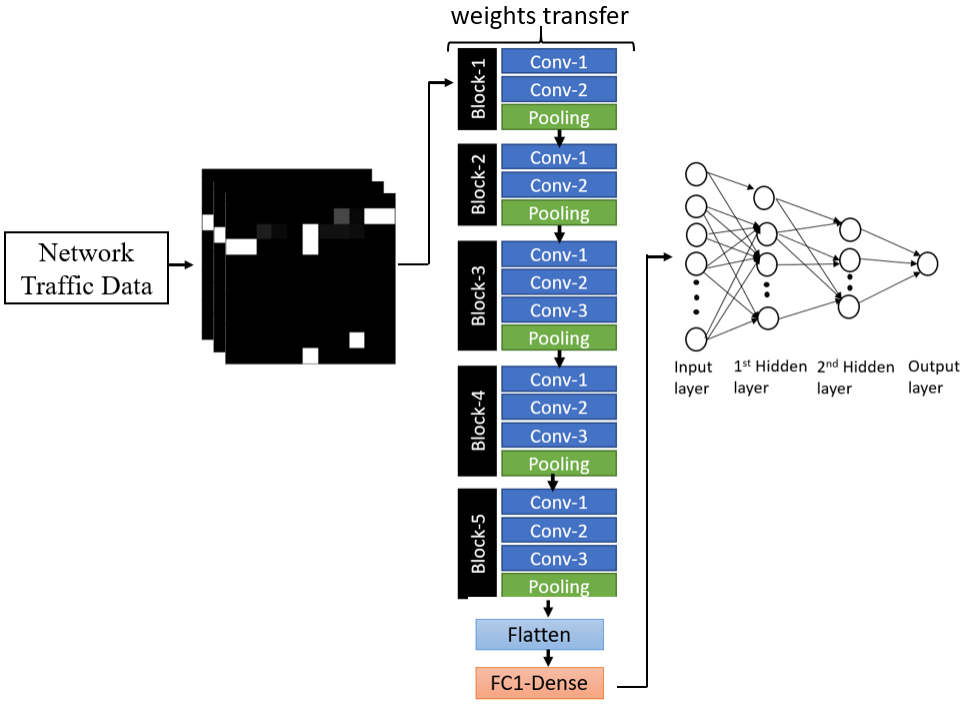

Figure 1. Architecture of proposed framework

Table 2. Sub-classes of different attacks

Types of Attacks	Sub-classes
DoS	apache2, back, land, Neptune, mailbomb, pod, processtable, smurf, teardrop, udstorm, and worm
Probe	ipsweap, mscan, nmap, portsweep, saint, and satan
U2R	buffer_overflow, loadmodule, perl, ps, rootkit, sqlattack, and xterm
R2L	ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, spy, snmguess, warezclient, warezmaster, xlovk, and xsnoop

(normal or attack). Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L) are four different types of attacks, where each attack includes several subclasses- DoS, Probe, U2R, and R2L contain 11, 6, 7, and 15 subclasses, respectively.

Table 2 exhibits subclasses of different attacks. The features in the dataset are divided into groups: intrinsic, content, host-and time-based features. The intrinsic features contain necessary information about the packet derived from the header of the connection, while the content features comprise information related to the original packets. Time-based features include information about the traffic input over a two-second window. On the other hand, host-based components contain information over a series of connections, designed to access attacks that are longer than a two-second window. Table 3 displays the distribution of normal and attacks data. Moreover, table 3 shows the distribution of data for different types of attacks: DoS, Probe, U2R, and R2L for the three datasets. We used 10% of each of the datasets to avoid computational complexity.

Table 3. Distributions of the NSL-KDD data

Type	KDDTrain+	KDDTest+	KDDTest-21
Normal	6641	973	199
Attack	5956	1281	986
DoS	4627	754	322
Probe	1233	283	256
U2R	87	237	199
R2L	9	7	5
Total Attacks	5956	1281	986

4.2. Data Preprocessing

We applied the NSL-KDD dataset that contains network connection features to evaluate the DCNN framework. Thus, data preprocessing is required to convert the raw data into image format to feed it to the

DCNN. The categorical data in NSL-KDD should be mapped to numeric data at first and then the overall data should be normalized. protocol type, flag, and service are the three categorical features in the NSL-KDD that are converted into numeric data using the one-hot encoder technique. We applied the min-max normalization technique to NSL-KDD to scale the original data to a fixed range of 0 and 1. The normalization ensures consistency of the data distribution and avoiding the exploding gradients problem in the training phase [3]. Equation (1) represents the min-max normalization formula, where X_{scaled}, and X is the normalized, and original value, respectively. $\min(X)$, and $\max(X)$ are the minimum and maximum values of the data.

$$X_{scaled} = \frac{X - \min(x)}{\max(x) - \min(x)} \quad (1)$$

The number of features has been expanded from 41 to 121 after the preprocessing step. To feed to the DCNN the 1-dimensional data is translated into a 2-d array of size 11×11 or grayscale image at first. Fig shows some network connection samples as grayscale images where images in the 1st row and 2nd row are for normal, and attack connections, respectively. In the next stage, the grayscale images were converted into 3-d RGB images and resized to the format of $224 \times 224 \times 3$ format as required by the pre-trained VGG-16 input shape. Since the grayscale images have only one-channel, in the process of conversion, a channel augmentation is performed by duplicating the grayscale images into three channels RGB images.

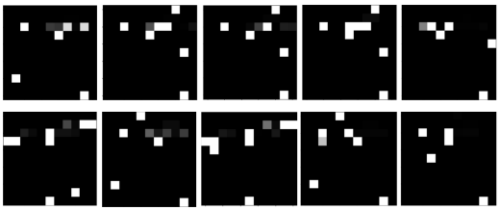

Figure 2. Grayscale images of network connection samples

4.3. Model evaluation metrics

To evaluate the models' performance, we considered accuracy, precision, recall, false alarm, and F-score metrics. These metrics use properties from the confusion matrix such as true positive (TP), false positive (FP), false negative (FN), and true negative (TN). TP is the number of attacks that are correctly classified as attacks, while FN is the attacks that are incorrectly classified. The number of incorrectly classified normal data is FN, and TN is correctly classified as normal data. Equation 2,3,4,5, and 6 are the mathematical definition of the performance metrics accuracy, precision, recall, false alarm, and F-score, respectively.

$$Accuracy = \frac{TP + TN}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$\mathrm{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\mathrm{False\ Alarm} = \frac{FP}{FP + TN} \quad (5)$$

$$\mathrm{F-score} = \frac{2 * Precision * recall}{Precision + recall} \quad (6)$$

4.4. Experimental design and Results

We evaluated our model performance by comparing it with the performance of traditional machine learning algorithms such as LR, SVM, and DT methods. Trained data was used to train each of the models we experimented with while test data was used for evaluating the performance of the models. The SVM, LR, and DT classifiers were applied to the dataset for comparing results with our proposed framework. The algorithms were implemented using Python scikit-learn library with available hyperparameter options. 'rbf' (Radial Basis Kernel) were chosen for SVM, 'gini' index was chosen for DT, and L2 penalty was chosen for LR classifier

Our proposed framework is based on a pre-trained VGG-16 model and a DNN. The DNN is trained with the features that are extracted using the pre-trained VGG-16 model. The DNN consists of four layers: an input layer, two hidden layers, and an output layer. We used 'ReLu' activation function in the hidden layer and 'sigmoid' function in the output layer. 'Adam' and 'binary cross-entropy' were used for optimizer and loss function respectively. We implemented an early stopping method to stop training once the model performance stops improving on the test data. We selected validation loss to be monitored for early stopping and set minimum delta to $1e-4$ (checks minimum change in the monitored quantity to qualify as an improvement) and patience to 10 (checks number of epochs that produced the monitored quantity with no improvement after which training will be stopped). Mini-batch gradient descent was considered and a batch size of 64 was chosen to train the model. The initial learning rate was set to 0.001 with a decay of $1e-5$ in every epoch. The L^2 regularization technique was applied to the output of the hidden layer to prevent the network from

overfitting and the regularization parameter 'lambda' was set to 0.001. All the parameters and hyperparameters used in the model were optimized by grid search.

We also applied other pre-trained models like VGG-19, Inception, MobileNet, and ResNet-50 to extract features from the converted RGB images and employ a DNN on the extracted features. We dropped the final classification layer, i.e., the softmax layer, from the mentioned pre-trained models' architecture, added a sigmoid layer of one neuron in binary classification, and added a softmax layer of five neurons for multi-class classification. We followed an identical structure for the DNN to maintain consistency of performance.

4.5. Results

We applied nine different methods for the binary classification problem, on the NSL-KDD dataset including traditional machine learning methods−SVM, DT, LR, and RF; transfer learning approaches− VGG-16+DNN, VGG-19+DNN, Inception V3+DNN, MobilNet+DNN, and ResNet-50+DNN. Table 4 represents the results of the methods with two different test datasets KDDTest+, and KDDTest-21 considering binary classification. The transfer learning-based methods express that the pre-trained model was use for feature extraction and DNN applied for classification. For instance, VGG-16+DNN: VGG-16 was applied for feature extraction from the image data, and the DNN was applied to the extracted features for intrusion classification. The experimental results show that our presented method (VGG-16+DNN) achieved the highest accuracy 89.30%, and F-score 90.26% for network detection for KDDTest+ while the decision tree achieved the second maximum accuracy of 80.65. The highest precision rate and lowest false alarm rate were achieved by DT. The highest accuracy (70.9%), recall (82.15%), and F-score (82.48) were achieved by the presented method, whereas maximum precision was obtained by SVM for the KDDTest-21 dataset. Other feature extractors VGG-19, MobileNet, Inception, and ResNet-50 also provides good performance, close to the VGG-16 methods.

The Table 5 shows the results for each of the classes of multi-class classification of network intrusion. Table 6 represents the results for multi-class classification for both the datasets. The VGG-16 feature extractor outperform others in terms of accuracy. VGG-16 + DNN achieved 78.39% and 52.56% for KDDTest+, and KDDTest-21 datasets, respectively. VGG-19 and MobileNet provides reasonable performance as well.

Table 4. Comparison of results for binary classification

	KDDTest+					KDDTest-21				
Methods	Accuracy (%)	Precision (%)	Recall (%)	False Alarm (%)	F-score (%)	Accuracy (%)	Precision (%)	Recall (%)	False Alarm (%)	F-score (%)
SVM	65.08	68.93	71.32	43.33	70.11	55.44	97.63	46.75	5.1	63.23
DT	80.65	95.44	69.62	4.41	80.51	68.10	95.12	64.36	14.95	76.78
LR	80.25	91.49	72.33	9.06	80.79	57.72	89.16	55.09	30.37	68.10
RF	73.91	93.36	58.73	5.60	72.10	65.51	81.39	74.35	77.10	77.71
VGG-16+ DNN	**89.30**	**93.55**	**87.19**	**7.9%**	**90.26**	**81.77**	**81.03**	**96.49**	**14.57**	**88.09**
VGG-19 + DNN	86.60	83.52	92.16	9.35	87.63	81.43	83.26	93.72	27.63	88.18
Inception V3 + DNN	85.71	78.84	95.19	5.2	86.25	75.61	79.00	90.47	41.20	84.35
MobileNet + DNN	85.71	78.84	95.19	5.2	86.25	79.57	79.81	94.81	21.60	86.67
ResNet50 + DNN	84.29	76.03	95.39	4.83	84.62	83.12	99.89	83.19	100.0	90.78

Table 5. Classification results for each class using VGG-16 + DNN

	KDDTest+			KDDTest-21		
	precision	recall	f1-score	precision	recall	f1-score
Normal	0.81	0.92	0.86	0.57	0.74	0.64
DoS	0.78	0.95	0.86	0.40	0.81	0.54
Probe	0.71	0.62	0.66	0.52	0.55	0.54
U2R	0.00	0.00	0.00	0.00	0.00	0.00
R2L	0.00	0.00	0.00	0.00	0.00	0.00

Table 6. Comparison of results for multi-class classification

	KDDTest+ Accuracy (%)	KDDTest-21 Accuracy (%)
VGG-16 + DNN	**78.39**	**52.56**
VGG-19 + DNN	74.62	38.90
Inception V3 + DNN	68.90	32.91
MobileNet + DNN	72.98	50.98
ResNet50 + DNN	69.38	41.85

5. Conclusions

Computer network attacks are increasingly posing a serious security threat. It is essential to develop an automatic network intrusion detection solution to reduce the risks of malicious activities. Existing traditional methods and machine learning algorithms are not sufficiently effective for network intrusion detection problems. In this paper, we proposed a transfer learning-based framework for network intrusion detection that first extracts higher level features leveraging VGG-16 pre-trained on ImageNet dataset by transferring wights. We trained the presented framework on KDDTrain+ dataset and evaluated performance on KDDTest+, and KDDTest-21 datasets, respectively. We also evaluated and compared performance of the framework with other advanced transfer learning from pre-trained models like VGG-19, MobileNet, Inception V3, and ResNet 50. We also compared the results with traditional machine learning models− LR, SVM, and DT methods. The experimental results show that using VGG-16 in the feature extraction process outperforms other pre-trained models that were used for feature extraction in terms of accuracy, precision, recall, false alarm, and f1-score.

6. References

[1] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., and Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-5). IEEE.

[2] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv: 1704.04 861.

[3] Kumar, V., and Sangwan, O. P. (2012). Signature based intrusion detection system using SNORT. International Journal of Computer Applications and Information Technology, 1(3), 35-41.

[4] Buczak, A. L., and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications surveys and tutorials, 18(2), 1153-1176.

[5] Xie, Y., and Richmond, D. (2018). Pre-training on grayscale ImageNet improves medical image classification. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 0-0).

[6] Wu, P., Guo, H., and Buckland, R. (2019, March). A transfer learning approach for network intrusion detection. In 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA) (pp. 281-285). IEEE.

[7] Wu, K., Chen, Z., and Li, W. (2018). A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access, 6, 50850-50859.

[8] Liu, P. (2019, February). An intrusion detection system based on convolutional neural network. In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering (pp. 62-67).

[9] Wu, P., Guo, H., and Buckland, R. (2019, March). A transfer learning approach for network intrusion detection. In 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA) (pp. 281-285). IEEE.

[10] Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE transactions on emerging topics in computational intelligence, 2(1), 41-50.

[11] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.

[12] Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[13] Kabir, E., Hu, J., Wang, H., and Zhuo, G. (2018). A novel statistical technique for intrusion detection systems. Future Generation Computer Systems, 79, 303-318.

[14] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). IEEE.

[15] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, Intrusion detection using convolutional neural networks for representation learning, Lect. Notes Comput. Sci. (including

Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10638 LNCS, pp. 858866, 2017.

[16] Masum, M., and Shahriar, H. (2019, December). Droid-NNet: Deep Learning Neural Network for Android Malware Detection. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 5789-5793). IEEE.

[17] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[18] Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., and De Geus, P. (2017, December). Malicious software classification using transfer learning of resnet-50 deep neural network. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1011-1014). IEEE.

[19] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

[20] Jaworek-Korjakowska, J., Kleczek, P., and Gorgon, M. (2019). Melanoma Thickness Prediction Based on Convolutional Neural Network with VGG-19 Model Transfer Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

